Integrated Modeling and Design of Nonlinear Control Systems

Gilmer L. Blankenship
Department of Electrical Engineering &
Institute for Systems Research
University of Maryland
College Park, MD 20742
gilmer@eng.umd.edu

Harry G. Kwatny
Department of Mechanical Engineering
& Mechanics
Drexel University
Philadelphia, PA 19104
hkwatny@coe.drexel.edu

Chris LaVigna
Techno-Sciences, Inc.
Suite 204
10001 Derekwood Lane
Lanham, MD 20706
chris@technosci.com

V. Polyakov
Department of Electrical Engineering &
Institute for Systems Research
University of Maryland
College Park, MD 20742
polyakov@pansy.umd.edu

Abstract: A summary description of a symbolic computing environment for nonlinear control system design is provided. The software includes capabilities for modeling multibody dynamics as well as linear and nonlinear control. Experience with applications is noted.

Introduction

The past two decades have seen dramatic theoretical advances in the analysis and control of nonlinear dynamical systems. However, the application of this conceptual progress to systems of meaningful scale has been impeded by the absence of good computational tools. One important reason is that the required calculations do not fit naturally into familiar numerical processes. Our view is that symbolic computing has matured to a point that does enable the application of modern nonlinear control theory to significant engineering problems. In this paper we describe software that has been under continuous development and application over the past several years.

Our goal has been to develop an efficient set of portable software tools that facilitates control system design from conception to implementation. Consequently, we are concerned with modeling, simulation, control system design and real time implementation. Control system design in a symbolic environment requires, of course, models in a symbolic or literal form. Modeling is a primary issue in control systems engineering practice and the need for symbolic models raises new issues.

We have been strongly influenced by the requirements of specific applications of concern to us. Our priorities and software choices reflect the practical needs of those applications. The overall structure of the design environment as we currently use it is illustrated in Figure 1. Our major contributions are the modeling and control design software implemented in *Mathematica*. In this paper we will describe and illustrate those capabilities.

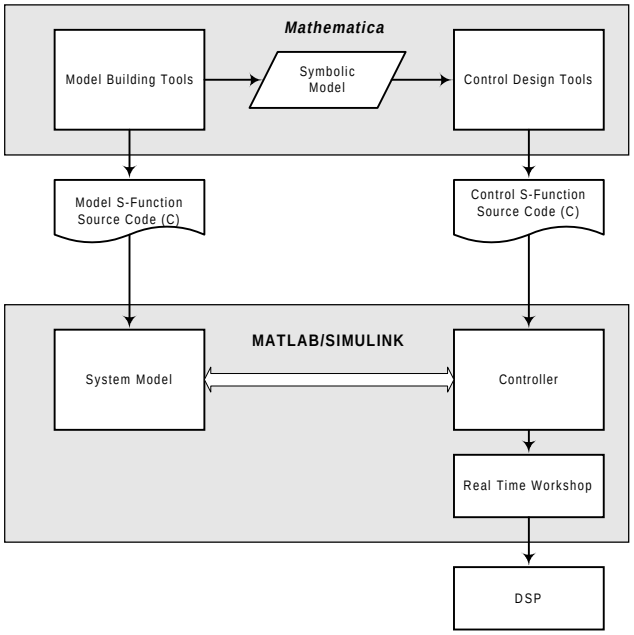

Figure 1. The control system design environment exploits the capabilities of *Mathematica* and MATLAB/SIMULINK. It integrates tools for model building, simulation, control system design and real time implementation.

Software Environment

Overview

The software used for modeling and control system design is built around *Mathematica* and MATLAB/SIMULINK, exploiting various packages and toolboxes that extend and integrate their basic capabilities. The symbolic computing tools we have implemented in *Mathematica* enable the efficient assembly of mathematical models for multibody dynamics [1] and provide for the design of nonlinear and adaptive controllers [2]. This

0-7803-3835-9/97/$10.00 (c) 1997 AACC

functionality is provided by the *Mathematica* packages TSi Dynamics and TSi Controls. These packages furnish several links to MATLAB/SIMULINK. TSi Dynamics can produce C MEX-files that define SIMULINK S-functions for simulation purposes [3], TSi Controls can produce C MEX-files that define SIMULINK S-functions for simulation or real time control implementation, and TSi Controls can provide linear (numerical) models for design and analysis using MATLAB's extensive facilities.

Symbolic Computing Capabilities

Modeling

In its current form the software contains a fairly comprehensive set of tools for assembling models of multibody mechanical systems. A few of these are listed in Table 1. The model building process has two distinctive features. First, the joints are defined in terms of their primitive action parameters from which all the required kinematic relations are derived. Thus, a user can contrive unusual joint configurations and is not restricted to a predefined set of standard joints. Second, the equations are formulated in Poincaré's form of Lagrange's equations that admits the standard Lagrange equations as a special case. However, Poincaré's form allows the exploitation of quasi–velocities which can greatly simplify the equations of motion.

The explicit models generated are of the form:

Kinematics: $\dot{q} = V(q)\,p$

Dynamics: $M(q)\,\dot{p} + C(q,p)\,p + F(q,p,u) = 0$

where q is a vector of configuration coordinates, p is a vector of quasi–velocities and u is a vector of exogenous inputs. They may be subjected to further symbolic processing for purposes such as nonlinear model reduction, nonlinear control system design or linearization. They may also be used for simulation or other numerical analysis procedures. To facilitate the latter applications, the package provides a direct interface to MATLAB/SIMULINK. In view of the complexity of models incorporating fully nonlinear kinematics, the C–code generated for this purpose, is organized to minimize the required numerical calculations.

To build a model, a user supplies defining data for individual joints and bodies, and the system structure. With this data, functions are available that can compute the kinetic energy function and inertia matrix as well as the gravitational potential energy function. It can also compute the strain potential energy and dissipation functions associated with deformations of flexible bodies. Various kinematic quantities can be obtained as well, e.g., end–effector configuration as a function of joint and deformation parameters. To complete a dynamic analysis, the user must supply the remaining parts of the potential energy function and definitions for any generalized forces. Functions are available to assist in developing these quantities.

Table 1. **Multibody Dynamics**

Function Name	Operation
Joints	returns all of the kinematic quantities corresponding to a list of joint definitions
TreeInertia	computes the inertia matrix of a multibody system in a tree structure containing flexible and rigid bodies
EndEffector	returns the Euclidean Configuration Matrix of a body fixed frame at a specified node
EndEffectorVelocity	returns the (6 dim) spatial velocity vector of a body fixed frame at a specified node
GeneralizedForce	computes the generalized force at specified node in terms of generalized coordinates
KinematicReplacements	sets up temporary replacement rules for repeated groups of expressions to simplify kinematic quantities
CreateModel	builds the kinematic and dynamic equations for tree structures
DiffConstrainedSys	adds differential constraints to a tree configuration
AlgConstrainedSys	adds algebraic constraints to a tree configuration

Control

The control software can be grouped into three general categories: linear control, nonlinear control and geometric tools. While our main interest has been in the development of software for nonlinear control system design we have found it convenient to have some linear control capabilities in the symbolic environment. Software tools are provided for the manipulation of linear controls systems in state space or frequency domain forms. Functions for the conversion of one form to the other are also included. Examples of the functions provided are listed in Table 2 and Table 3.

We have also implemented functions required to apply modern geometric methods of control system design to nonlinear affine systems [4, 5]. These methods play an important role in

adaptive control system design [2, 5-7] and variable structure control as well [8, 9]. Typical functions are given in Table 4 and Table 5. Table 6 lists examples of geometric functions that have been developed to support the control analysis constructions.

Table 2. **Linear Systems: State Space**

Function Name	Operation
Controllable/Observable	tests for controllability and observability
ControllabilityMatrix **ObservabilityMatrix**	returns the controllability or observability matrices, respectively
PolePlace	state feedback pole placement based on Ackermann's formula with options
DecouplingConrol	state feedback and coordinate transformation that decouples input-output map
RelativeDegree	computes the vector relative degree
LQR, LQE	compute optimal quadratic regulator and estimator parameters

Table 3. **Linear Systems: Frequency Domain**

Function Name	Operation
LeastCommonDenominator	finds the least common denominator of the elements of a proper, rational G(s)
Poles	finds the roots of the least common denominator
LaurentSeries	computes the Laurent series up to specified order
HankelMatrix	computs the Hankel matrix associated with Laurent expansion of G(s)
McMillanDegree	computes the degree of the minimal realization of G(s)
ControllableRealization **ObservableRealization**	compute, respectively, the controllable and observable realizations of a transfer function

Table 4. **Nonlinear systems: Geometric Control**

Function Name	Operation
VectorRelativeOrder	computes the relative degree vector
DecouplingMatrix	computes the decoupling matrix
IOLinearize	computes the linearizing control
NormalCoordinates	computes the partial state transformation,
LocalZeroDynamics	computes the local form of the zero dynamics
StructureAlgorithm	computes the parameters of an inverse system
DynamicExtension	applies dynamic extension as a remedy for singular decoupling matrix

Table 5. **Nonlinear systems: Adaptive Control**

Function Name	Operation
AdaptiveRegulator	generates an adaptive regulator for a class of linearizable systems
AdaptBackstepReg	computes an adaptive regulator by backstepping for SISO systems in PSFF form
AdaptiveTracking	computes an adaptive tracking controller
PSFFCond	tests a system to determine if it is reducible to PSFF form
PSFFSolve	transforms a system to PSFF form if possible

Table 6. **Nonlinear systems: Geometric Tools**

Function Name	Operation
LieBracket	computes the Lie bracket of a given pair of vector fields
Ad	computes the iterated Lie bracket of specified order of a pair of vector fields
Involutive	tests a set of vector fields to determine if it is involutive
Span	generates a set of basis vector fields for a given set of vector fields
FlowComposition	generates a composite function from a given set of flows
ParametricManifold	computes a parametric representation for an imbedded manifold
StateTransformation	transforms nonlinear dynamic models in various forms

Example

As an illustration of integrated modeling and control we will consider the application of the symbolic computing tools to the lateral dynamics of an automobile as illustrated in Figure 2. The problem is simple enough that we can present results in limited space. The vehicle has only three degrees of freedom associated with the coordinates (x, y, θ). It has three controls: steering angle δ, drive forces F_r, F_l at the right and left wheel, which can be represented in terms of an $F, \delta F$ defined via $F_r = F + \delta F$ and $F_l = F - \delta F$. Thus, we have three independent controls $\delta, F, \delta F$. δF could be used for traction control, but we will not consider that here.

We consider the possibility of shaping the lateral handling qualities by using a stability augmenting regulator that

manipulates δ and F to control speed V_s and angular velocity ω. First, we consider modeling and then control.

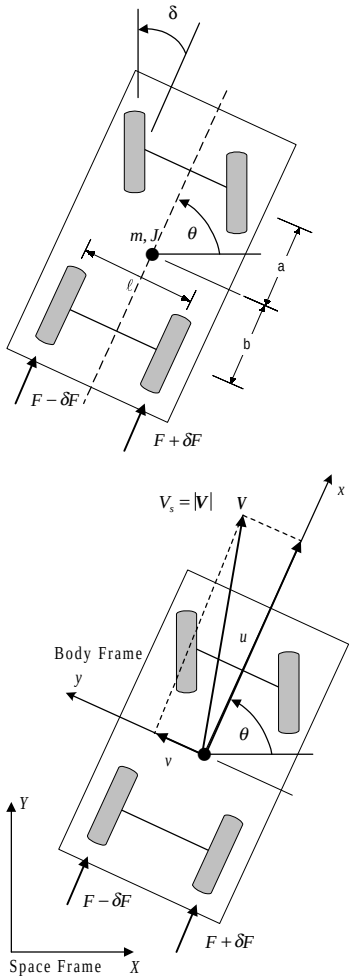

Figure 2. A simple model for the lateral dynamics of an automobile, patterned after [10].

Modeling

We construct a model similar to that described in [10] (see also [11]) except that, unlike the linear model developed therein, we retain all of the geometric nonlinearities. We also include caster and camber in the front wheels. We will present results, however, in which camber is assumed to be negligible and caster is small so that it can be approximated by first order terms.

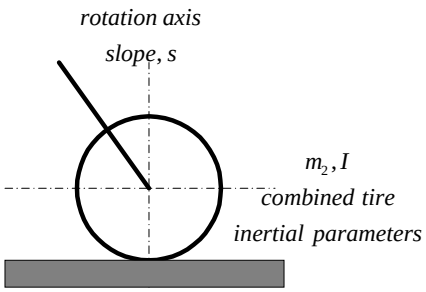

Figure 3. Physical characteristics of the front wheels. The tire side force coefficient is κ.

We do not consider the suspension system, the rotational energy of the wheels, aerodynamics, or other details of the steering system that might be of interest. One important aspect of computer model construction is that it is relatively easy to rebuild the model (and simulation code for that matter) with such refinements.

Modeling proceeds as follows:

- define the joints
- define the body geometry and inertial parameters
- define the system interconnection structure
- define the potential energy storage components
- define and develop the generalized forces

The model can then be assembled. If desired the model can be further manipulated, for example by a coordinate transformation. In this case, it is convenient to transform the linear velocity representation from components v_x, v_y to speed and sideslip angle V_s, β: $v_x = V_s \cos\beta$, $v_y = V_s \sin\beta$. Thus, we obtain:

$$\begin{bmatrix} J_{zz} + 2I_{zz} + 2(a^2+b^2)m_2 & 2am_2 \sin\beta & 2am_2 V_s \cos\beta \\ 0 & (m+2m_2)\cos\beta & -(m+2m_2)V_s \sin\beta \\ 2am_2 & (m+2m_2)\sin\beta & (m+2m_2)V_s \cos\beta \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \dot{V}_s \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

where:

$$f_1 = -\frac{2}{V_s^2}\left(\left(\frac{\ell}{2}\right)^2 \delta\kappa\omega(\beta V_s + a\omega) + V_s\left(a^2\kappa\omega + \kappa\left(-b\beta V_s + \beta R s V_s + b^2\omega\right) + a\left(\beta\kappa V_s - \delta\kappa V_s\right) + 2\kappa R s\omega + m_2 V_s^2\omega\right)\right)$$

$$f_2 = 2F + 2\beta\delta\kappa + \frac{2a\delta\kappa\omega}{V_s} + \beta(m+2m_2)V_s\omega + 2am_2\omega^2$$

$$f_3 = \frac{1}{V_s}\left(-4\beta\kappa V_s + 2\delta\kappa V_s - \left(2a\kappa - 2b\kappa + 2\kappa R s + mV_s^2 + 2m_2 V_s^2\right)\omega\right)$$

The functions f_1, f_2, f_3 have been simplified for presentation by expanding them to first order in s, δ and second order in β, ω.

I-O Linearization

The goal of a driver the vehicle is to regulate V_s, ω using the controls F, δ. A stability augmenting regulator can be used to shape the dynamics as seen by the driver. The design of such a regulator can be approached via feedback linearizaton. The idea is to linearize and decouple the input-output dynamics and then employ a stabilizing linear compensator. For this approach to be viable, the system must have well-defined vector relative degree and the zero dynamics must be stable. It is a straightforward matter to use the functions in Table 4 to compute the vector relative degree, the feedback linearizing control and the zero dynamics.

To perform these computations requires:

- place the system in state variable form: $\dot{x} = F(x,u)$

- separate the right hand side into the form: $f(x)+G(x)u$
- compute the feedback linearizing control
- compute the normal coordinates
- compute the zero dynamics

The calculations reveal that the vector relative degree is {1,1}, as anticipated, so that the zero dynamics are of dimension one. The linearizing control is quite complex so it is not displayed here. A local description of the zero dynamics is obtained around an equilibrium point corresponding to straight motion with constant velocity V_0. All parameters have been specified except tire radius *R* and camber *s*. Up to third order the zero dynamics are:

$$\dot{w}_1 = \frac{-171.243 + 18.5128Rs}{V_0} w_1 + \frac{-85.6216 + 9.25639Rs}{V_0} w_1^3$$

Concluding Remarks

The authors and colleagues have applied these tools to several more substantive modeling and control problems. Modeling and simulation applications include a 15 degree of freedom tracked vehicle with flexible hull and 10 road wheels [12], an 18 wheel tractor-trailer, and numerous robot configurations with flexible and rigid elements. Modeling and control applications include a 30 mm Apache chain gun with flexible barrel and novel sensing and actuation devices [13], [14]. That project enabled the full scope of the tools to be demonstrated including model validation and real time control.

Ongoing work in flight dynamics and control has led to new symbolic tools for generating parametrically dependent equilibrium surfaces and deriving linear families of models from parameter dependent nonlinear dynamics [15]. Symbolic computing tools for modeling and adaptive control for systems with hard (nondifferentiable) nonlinearities [16] are under development and real time implementation of adaptive controllers is planned.

References

[1] H. G. Kwatny and G. L. Blankenship, "Symbolic Construction of Models for Multibody Dynamics," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 271-281, 1995.

[2] G. L. Blankenship, R. Ghanadan, H. G. Kwatny, C. LaVigna, and V. Polyakov, "Integrated tools for Modeling and Design of Controlled Nonlinear Systems," *IEEE Control Systems*, vol. 15, pp. 65-79, 1995.

[3] Anonomous, *SIMULINK: Release Notes V1.3*. Natick: The MathWorks, Inc., 1994.

[4] A. Isidori, *Nonlinear Control Systems*, 3 ed. London: Springer-Verlag, 1995.

[5] H. Nijmeijer and H. J. van der Schaft, *Nonlinear Dynamical Control Systems*. New York: Springer–Verlag, 1990.

[6] I. Kanellakapoulos, P. V. Kokotovic, and A. S. Morse, "Systematic design of Adaptive Controllers for Feedback Linearizable Systems," *IEEE Transactions on Automatic Control*, vol. AC–36, pp. 1241–1253, 1991.

[7] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, "Adaptive Nonlinear Control Without Overparameterization," *Systems and Control Letters*, vol. 19, pp. 177-185, 1992.

[8] H. G. Kwatny and H. Kim, "Variable Structure Regulation of Partially Linearizable Dynamics," *Systems & Control Letters*, vol. 15, pp. 67–80, 1990.

[9] H. G. Kwatny and G. L. Blankenship, "Symbolic Tools for Variable Structure Control System Design: The Zero Dynamics," presented at IFAC Symposium on Robust Control via Variable Structure and Lyapunov Techniques, Benevento, Italy, 1994.

[10] E. O. Doebelin, *System Modeling and Response: Theoretical and Experimental Approaches*. New York: John Wiley & Sons, 1980.

[11] J. I. Neimark and N. A. Fufaev, *Dynamics of Nonholonomic Systems*, vol. 33. Providence: American Mathematical Society, 1972.

[12] C. LaVigna, H. G. Kwatny, and G. L. Blankenship, "Flexible Multibody Dynamical Analysis System," Techno–Sciences, Inc., Lanham Phase I Final Report, Contract No. DAAE07–93–CR022, December 1993.

[13] C. Lavigna, "Adaptive Stabilization of Weapon Systems Using Surface Mounted Adaptive Structure Technology," Techno-Sciences, Inc., Lanham, Phase II Final Report, DAAA21-94-C-0066 November 1996.

[14] C. LaVigna, H. G. Kwatny, G. L. Blankenship, and M. Mattice, "A Rapid Prototyping Workstation for Design of Distributed Parameter Control Systems," in *Computational Science in the 21st Century*, J. Periaux, Ed.: John Wiley & Sons, 1997.

[15] H. G. Kwatny and B.-C. Chang, "Constructing Linear Families from Parameter-Dependent Nonlinear Dynamics," *to be submitted*, 1997.

[16] G. Tao and P. V. Kokotovic, *Adaptive Control of Systems with Actuator and Sensor Nonlinearities*. New York: J. Wiley and Sons, Inc., 1996.

0-7803-3835-9/97/$10.00 (c) 1997 AACC